

ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE PIPMOD

OBJECT MODULE PLACED IN PIP.OBJ

COMPILER INVOKED BY: IF0: PIP.PLM DATE(MARCH 16, 1982) XREF OPTIMIZE(3) DEBUG

```

1      $title ('PERIPHERAL INTERCHANGE PROGRAM')
      PIPMOD:
      DO;
/* PERIPHERAL INTERCHANGE PROGRAM

```

```

      COPYRIGHT (C) 1976, 1977, 1978, 1979, 1980, 1981
      DIGITAL RESEARCH
      BOX 579
      PACIFIC GROVE, CA
      93950

```

Revised:

```

      17 Jan 80 by Thomas Rolander (MP/M 1.1)
      05 Oct 81 by Ray Pedrizetti (MP/M-86 2.0)
      18 Dec 81 by Ray Pedrizetti (CP/M-86 1.1) 2/

```

```

/* Command lines used for CMD file generation */

```

```

/* (on VAX)
asm86 scd.a36
plm86 pip.plm debug xref optimize(3) date(1/8/82)
link86 scd1.obj,inpout.obj,pip.obj,plm86.lib to pip.lnk
loc86 pip.lnk od(sm(code,dats,data,stack,const)) -
      od(sm(code(0))) ss(stack(+32))
h86 pip

(on a micro)
vax pip.h86 $fans
gencmd pip data[b17b n230 x880]

```

```

      * note the beginning of the data segment will change when
      * the program is changed. see the 'MP2' file generated
      * by LOC86. the constants are last to force hex generation
*/

```

```

2 1  DECLARE
      VERSION LITERALLY '0022H', /* REQUIRED FOR OPERATION */
      aversion literally '1130H'; /* for MP/M-86 operation */

3 1  DECLARE
      MAXB ADDRESS EXTERNAL, /* ADDR FIELD OF JMP DDOS */
      FCB (33) BYTE EXTERNAL, /* DEFAULT FILE CONTROL BLOCK */
      BUFF(128)BYTE EXTERNAL; /* DEFAULT BUFFER */

4 1  DECLARE
      ENDFILE LITERALLY '1AH'; /* END OF FILE MARK */

5 1  declare /* main program entry label */
      plm label public;

```

CP/M-86 v.1.1 (Vol. II)
 COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # C81-162

' (1/8/82) CP/M-86 PIP VERS 1.1 ');

```

/* LITERAL DECLARATIONS */
7 1 DECLARE
  LIT LITERALLY 'LITERALLY',
  LPP LIT '60', /* LINES PER PAGE */
  TAB LIT '09H', /* HORIZONTAL TAB */
  FF LIT '0CH', /* FORM FEED */
  LA LIT '05FH', /* LEFT ARROW */
  LB LIT '05BH', /* LEFT BRACKET */
  RB LIT '05DH', /* RIGHT BRACKET */

  FSIZE LIT '33',
  RSIZE LIT '36', /* SIZE OF RANDOM FCB */
  NSIZE LIT '3',
  FNSIZE LIT '11',
  FEXI LIT '7',
  FEXTL LIT '3',

  /* scanner return type code */
  outt LIT '0', /* output device */
  PRNT LIT '1', /* PRINTER */
  LSTT LIT '2', /* list device */
  axot LIT '3', /* auxiliary output device */
  FILE LIT '4', /* file type */
  CONS LIT '5', /* CONSOLE */
  axit LIT '6', /* auxiliary input device */
  inpt LIT '7', /* input device */
  MULT LIT '8', /* nul characters */
  EOFT LIT '9', /* EOF character */
  ERR LIT '10', /* error type */
  SPECL LIT '11', /* special character */
  DISKNAME LIT '12', /* diskname letter */

8 1 DECLARE
  SEARCH LITERALLY 'FCB'; /* SEARCH FCB IN MULTI COPY */

9 1 DECLARE
  TRUE LITERALLY '1',
  FALSE LITERALLY '0',
  FOREVER LITERALLY 'WHILE TRUE',
  CR LITERALLY '13',
  LF LITERALLY '10',
  WHAT LITERALLY '63';

10 1 DECLARE
  COLUMN BYTE, /* COLUMN COUNT FOR PRINTER TABS */
  LINENO BYTE, /* LINE WITHIN PAGE */
  FEEDBASE BYTE, /* USED TO FEED SEARCH CHARACTERS */
  FEEDLEN BYTE, /* LENGTH OF FEED STRING */
  MATCHLEN BYTE, /* USED IN MATCHING STRINGS */
  QUITLEN BYTE, /* USED TO TERMINATE QUIT COMMAND */
  CDISK BYTE, /* CURRENT DISK */
  SBLEN ADDRESS, /* SOURCE BUFFER LENGTH */
  DBLEN ADDRESS, /* DEST BUFFER LENGTH */
  tblen address, /* temp buffer length */

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

SBASE  ADDRESS, /* SOURCE BUFFER BASE */

/* THE VECTORS DBUFF AND SBUFF ARE DECLARED WITH DIMENSION
1024, BUT ACTUALLY VARY WITH THE FREE MEMORY SIZE */
DBUFF(1024) BYTE AT (.MEMORY), /* DESTINATION BUFFER */
SBUFF BASED SBASE (1024) BYTE, /* SOURCE BUFFER */

    /* source fcb, password and password mode */
source structure (
    fcb(fsize) byte,
    user      byte,
    type      byte ),

    /* temporary destination fcb, password and password mode */
dest structure (
    fcb(fsize) byte,
    user      byte,
    type      byte ),

    /* original destination fcb, password and password mode */
odest structure (
    fcb(fsize) byte,
    user      byte,
    type      byte ),

filesize(3) byte, /* file size random record number */

DESTR  ADDRESS AT(.DEST.FCB(33)), /* RANDOM RECORD POSITION */
SOURCER ADDRESS AT(.SOURCE.FCB(33)), /* RANDOM RECORD POSITION */
DESTR2 BYTE AT(.DEST.FCB(35)), /* RANDOM RECORD POSITION R2 */
SOURCER2 BYTE AT(.SOURCE.FCB(35)), /* RANDOM RECORD POSITION R2 */

extsave byte, /* temp extent byte for bdos bug */

nsbuf  address, /* next source buffer */
NSOURCE ADDRLESS, /* NEXT SOURCE CHARACTER */
NDEST  ADDRESS; /* NEXT DESTINATION CHARACTER */

```

```

11 1  DECLARE
    fastcopy byte, /* true if copy directly to dbuf */
    dblbuf  byte, /* true if both source and dest buffer used */
    concat  byte, /* true if concatenation command */
    aabig   byte, /* true if file is aabig type */
    dfile   byte, /* true if dest is file type */
    sfile   byte, /* true if source is file type */
    mode    byte, /* true if file already mode */
    endofsrc byte, /* true if end of source file */
    nendcmd byte, /* true if not end of command tail */
    insparc byte, /* true if in middle of sparse file */
    sparfil byte, /* true if sparse file being copied */
    MULTCOM BYTE, /* FALSE IF PROCESSING ONE LINE */
    PUTNUM  BYTE, /* SET WHEN READY FOR NEXT LINE NUM */
    CONCNT  BYTE, /* COUNTER FOR CONSOLE READY CHECK */
    CHAR    BYTE, /* LAST CHARACTER SCANNED */
    FLEN    BYTE; /* FILE NAME LENGTH */

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

f1    byte,      /* f1 user attribute flag */
f2    byte,      /* f2 user attribute flag */
f3    byte,      /* f3 user attribute flag */
f4    byte,      /* f4 user attribute flag */
ro    byte,      /* read only attribute flag */
sys   byte,      /* system attribute flag */
dcnt  byte;      /* error code or directory code */

```

```

13 1  DECLARE CBUF(130) BYTE, /* COMMAND BUFFER */
      MAXLEN  BYTE AT (.CBUF(0)), /* MAX BUFFER LENGTH */
      COMLEN  BYTE AT (.CBUF(1)), /* CURRENT LENGTH */
      COMBUF(128) BYTE AT (.CBUF(2)); /* COMMAND BUFFER CONTENTS */

```

```

14 1  DECLARE
      CBP BYTE; /* COMMAND BUFFER POINTER */

```

```

15 1  DECLARE
      CUSER BYTE; /* CURRENT USER NUMBER */

```

```

16 1  declare
      last$user byte;

```

```

17 1  DECLARE /* CONTROL TOGGLE VECTOR */
      CONT(26) BYTE, /* ONE FOR EACH ALPHABETIC */
      /* 00 01 02 03 04 05 06 07 08 09 10 11 12 13
         A  B  C  D  E  F  G  H  I  J  K  L  M  N
         14 15 16 17 18 19 20 21 22 23 24 25
         O  P  Q  R  S  T  U  V  W  X  Y  Z */
      archiv byte at(.cont(0)), /* file archive */
      DELET  BYTE AT(.CONT(3)), /* DELETE CHARACTERS */
      ECHO  BYTE AT(.CONT(4)), /* ECHO CONSOLE CHARACTERS */
      FORMF BYTE AT(.CONT(5)), /* FORM FILTER */
      GETU  BYTE AT(.CONT(6)), /* GET FILE, USER # */
      HEXT  BYTE AT(.CONT(7)), /* HEX FILE TRANSFER */
      IGNOR BYTE AT(.CONT(8)), /* IGNORE :00 RECORD ON FILE */
      KILDS byte at(.cont(10)), /* kill filename display */
      LOWER BYTE AT(.CONT(11)), /* TRANSLATE TO LOWER CASE */
      NUMB  BYTE AT(.CONT(13)), /* NUMBER OUTPUT LINES */
      OBJ   BYTE AT(.CONT(14)), /* OBJECT FILE TRANSFER */
      PAGCNT BYTE AT(.CONT(15)), /* PAGE LENGTH */
      QUIT  BYTE AT(.CONT(16)), /* QUIT COPY */
      RSYS  BYTE AT(.CONT(17)), /* READ SYSTEM FILES */
      STARTS BYTE AT(.CONT(18)), /* START COPY */
      TABS  BYTE AT(.CONT(19)), /* TAB SET */
      UPPER BYTE AT(.CONT(20)), /* UPPER CASE TRANSLATE */
      VERIF BYTE AT(.CONT(21)), /* VERIFY EQUAL FILES ONLY */
      WRROF BYTE AT(.CONT(22)), /* WRITE TO R/O FILE */
      ZEROP BYTE AT(.CONT(25)); /* ZERO PARITY ON INPUT */

```

```

18 1  DECLARE ZEROSUP BYTE, /* ZERO SUPPRESSION */
      (C3,C2,C1) BYTE; /* LINE COUNT ON PRINTER */

```

```

19 1  OUTD: PROCEDURE(B) external;
20 2  DECLARE B BYTE;
      /* SEND B TO OUT: DEVICE */
21 2  END OUTD;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```
22 1  INPD: PROCEDURE BYTE external;
23 2  END INPD;

24 1  MON1: PROCEDURE(F,A) EXTERNAL;
25 2  DECLARE F BYTE,
26 2  A ADDRESS;
27 2  END MON1;

27 1  MON2: PROCEDURE(F,A) BYTE EXTERNAL;
28 2  DECLARE F BYTE,
29 2  A ADDRESS;
30 2  END MON2;

30 1  MON3: PROCEDURE(F,A) ADDRESS EXTERNAL;
31 2  DECLARE F BYTE,
32 2  A ADDRESS;
33 2  END MON3;

33 1  BOOT: PROCEDURE;
34 2  /* SYSTEM REBOOT */
35 2  CALL MON1(0,0);
36 2  END BOOT;

36 1  RDCHAR: PROCEDURE BYTE;
37 2  /* READ CONSOLE CHARACTER */
38 2  RETURN MON2(1,0);
39 2  END RDCHAR;

39 1  PRINTCHAR: PROCEDURE(CHAR);
40 2  DECLARE CHAR BYTE;
41 2  CALL MON1(2,CHAR AND 7FH);
42 2  END PRINTCHAR;

43 1  CRLF: PROCEDURE;
44 2  CALL PRINTCHAR(CR);
45 2  CALL PRINTCHAR(LF);
46 2  END CRLF;

47 1  printx: procedure(a);
48 2  declare a address;
49 2  call mon1(7,a);
50 2  end printx;

51 1  PRINT: PROCEDURE(A);
52 2  DECLARE A ADDRESS;
53 2  /* PRINT THE STRING STARTING AT ADDRESS A UNTIL THE
54 2  NEXT DOLLAR SIGN IS ENCOUNTERED */
55 2  CALL CRLF;
56 2  CALL printx(A);
57 2  END PRINT;

56 1  RDCOM: PROCEDURE;
57 2  /* READ INTO COMMAND BUFFER */
58 2  MAXLEN = 128;
59 2  CALL MON1(10,MAXLEN);
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```
59 2      END RUCON;

60 1      CONBRK: PROCEDURE BYTE;
          /* CHECK CONSOLE CHARACTER READY */
61 2      RETURN MON2(11,0);
62 2      END CONBRK;

63 1      CVERSION: PROCEDURE ADDRESS;
64 2      RETURN MON3(12,0); /* VERSION NUMBER */
65 2      END CVERSION;

66 1      SETDMA: PROCEDURE(A);
67 2      DECLARE A ADDRESS;
68 2      CALL MON1(26,A);
69 2      END SETDMA;

70 1      OPEN: PROCEDURE(FCB);
71 2      DECLARE FCB ADDRESS;
72 2      dcnt = mon2(15,FCB);
73 2      END OPEN;

74 1      CLOSE: PROCEDURE(FCB);
75 2      DECLARE FCB ADDRESS;
76 2      dcnt = MON2(16,FCB);
77 2      END CLOSE;

78 1      ck$user: procedure;
79 2      do forever;
80 3          if dcnt = 0fff then return;
82 3          if last$user = buff(ror (dcnt,3) and 110$0000b)
              then return;
84 3          dcnt = mon2(18,0);
85 3      end;
86 2      end ck$user;

87 1      SEARCH: PROCEDURE(FCB);
88 2      DECLARE FCB ADDRESS;
89 2      dcnt = MON2(17,FCB);
90 2      call ck$user;
91 2      END SEARCH;

92 1      SEARCHN: PROCEDURE;
93 2      dcnt = MON2(18,0);
94 2      call ck$user;
95 2      END SEARCHN;

96 1      DELETE: PROCEDURE(FCB);
97 2      DECLARE FCB ADDRESS;
98 2      CALL MON1(19,FCB);
99 2      END DELETE;

100 1     DISKRD: PROCEDURE(FCB);
101 2     DECLARE FCB ADDRESS;
102 2     dcnt = MON2(20,FCB);
103 2     END DISKRD;

104 1     DISKWRITE: PROCEDURE(FCB);
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```
105 2    DECLARE FCB ADDRESS;
106 2    dcnt = MON2(21,FCB);
107 2    END DISKWRITE;

108 1    MAKE: procedure(fcba);
109 2    declare fcba address;
110 2    dcnt = mon2(22,fcba);
111 2    END MAKE;

112 1    RENAME: PROCEDURE(FCB);
113 2    DECLARE FCB ADDRESS;
114 2    dcnt = MON2(23,FCB);
115 2    END RENAME;

116 1    getdisk: procedure byte;
117 2    return mon2(25,0);
118 2    end getdisk;

119 1    SETIND: PROCEDURE(FCB);
120 2    DECLARE FCB ADDRESS;
121 2    dcnt = MON2(30,FCB);
122 2    END SETIND;

123 1    GETUSER: PROCEDURE BYTE;
124 2    RETURN MON2(32,OFFH);
125 2    END GETUSER;

126 1    SETUSER: PROCEDURE(USER);
127 2    DECLARE USER BYTE;
128 2    CALL MON1(32,(last:user:=USER));
129 2    END SETUSER;

130 1    SETCUSER: PROCEDURE;
131 2    CALL SETUSER(CUSER);
132 2    END SETCUSER;

133 1    setduser: procedure;
134 2    call setuser(odest.user);
135 2    end setduser;

136 1    SETSUSER: PROCEDURE;
137 2    CALL SETUSER(source.user);
138 2    END SETSUSER;

139 1    RD$RANDOM: PROCEDURE(FCB) BYTE;
140 2    DECLARE FCB ADDRESS;
141 2    RETURN MON2(33,FCB);
142 2    END RD$RANDOM;

143 1    write$random: procedure(fcb) byte;
144 2    declare fcb address;
145 2    return mon2(34,fcb);
146 2    end write$random;

147 1    retfsize: procedure(fcb) byte;
148 2    declare fcb address;
149 2    return mon2(35,fcb);
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

150 2      end retfsize;

151 1      SET$RANDOM: PROCEDURE(FCB);
152 2      DECLARE FCB ADDRESS;
          /* SET RANDOM RECORD POSITION */
153 2      CALL MON1(36,FCB);
154 2      END SET$RANDOM;

155 1      MOVE: PROCEDURE(S,D,N);
156 2      DECLARE (S,D) ADDRESS, N BYTE;
157 2      DECLARE A BASED S BYTE, B BASED D BYTE;
158 2      DO WHILE (N:=N-1) <> 255;
159 3      B = A; S = S+1; D = D+1;
162 3      END;
163 2      END MOVE;

164 1      ERROR: PROCEDURE(errtype,fileadr);
165 2      DECLARE I BYTE,
          temp byte,
          errtype byte,
          fileadr address,
          fcb based fileadr (fsize) byte;

          /* errtype error messages */
166 2      declare er00(*) byte data ('DISK READ$');
167 2      declare er01(*) byte data ('DISK WRITE$');
168 2      declare er02(*) byte data ('VERIFY$');
169 2      declare er03(*) byte data ('INVALID DESTINATION$');
170 2      declare er04(*) byte data ('INVALID SOURCE$');
171 2      declare er05(*) byte data ('USER ABORTED$');
172 2      declare er06(*) byte data ('BAD PARAMETER$');
173 2      declare er07(*) byte data ('INVALID USER NUMBER$');
174 2      declare er08(*) byte data ('INVALID FORMAT$');
175 2      declare er09(*) byte data ('HEX RECORD CHECKSUM$');
176 2      declare er10(*) byte data ('FILE NOT FOUND$');
177 2      declare er11(*) byte data ('START NOT FOUND$');
178 2      declare er12(*) byte data ('QUIT NOT FOUND$');
179 2      declare er13(*) byte data ('INVALID HEX DIGITS$');
180 2      declare er14(*) byte data ('CLOSE FILE$');
181 2      declare er15(*) byte data ('UNEXPECTED END OF HEX FILE$');
182 2      declare er16(*) byte data ('INVALID SEPARATORS$');
183 2      declare er17(*) byte data ('NO DIRECTORY SPACE$');
184 2      declare er18(*) byte data ('INVALID FORMAT WITH SPARSE FILE$');

185 2      declare errmsg(*) address data(
          .er00,.er01,.er02,.er03,.er04,
          .er05,.er06,.er07,.er08,.er09,
          .er10,.er11,.er12,.er13,.er14,
          .er15,.er16,.er17,.er18);

186 2      call setduser;
187 2      if made then
188 2      do; call close(.dest);
190 3      call delete(.dest); /* delete destination scratch file */
191 3      end;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____


```

/* print out error message */
192 2    call print(,('ERROR: $'));
193 2    call printx(errmsg(errtype));
194 2    call printx(,(' - $'));
195 2    if fileadr < 0 then
196 2        do; call printchar('A' + fcb(0) - 1);
197 3        call printchar(':');
198 3        do i = 1 to fsize;
199 4            if (temp := fcb(i) and 07fh) < ' ' then
200 4                do; if i = fext then call printchar(' ');
201 4                    call printchar(temp);
202 5                end;
203 4            end;
204 3        end;
205 3    end;

/* ZERO THE COMLEN IN CASE THIS IS A SINGLE COMMAND */
208 2    COMLEN = 0;
209 2    CALL CRLF;
210 2    GO TO RETRY;
211 2    END ERROR;

FORMERR: PROCEDURE;
212 1    call error(8,0); /* invalid format */
213 2    END FORMERR;

maxsize: procedure byte;
214 1    if (source.fcb(35) = filesize(2)) and
215 2        (source.fcb(34) = filesize(1)) and
216 2        (source.fcb(33) = filesize(0)) then return true;
217 2    return false;
218 2    end maxsize;

SETUPDEST: PROCEDURE;
219 1    call setduser; /* destination user */
220 2    call move(.odest,.dest,(fsize + 1)); /* save original dest */
221 2    /* MOVE THREE CHARACTER EXTENT INTO DEST FCB */
222 2    CALL MOVE(,('$$$'),.DEST.FCB(FEXT),FEXTL);
223 2    CALL DELETE(.DEST); /* REMOVE OLD $$$ FILE */
224 2    CALL MAKE(.DEST); /* CREATE A NEW ONE */
225 2    IF DCNT = 255 THEN
226 2        call error(17,.dest); /* no directory space */
227 2    DEST.FCB(32) = 0;
228 2    made = true;
229 2    END SETUPDEST;

SETUPSOURCE: PROCEDURE;
230 1    CALL SETUSER; /* SOURCE USER */
231 2    CALL OPEN(.SOURCE); /* open source */
232 2    IF (NOT RSYS) AND ROL(SOURCE.FCB(10),1) THEN
233 2        /* skip system file */
234 2        DCNT = 255;
235 2    IF DCNT = 255 THEN
236 2        call error(10,.source); /* file not found */
237 2        f1 = source.fcb(1) and 80h; /* save file attributes */
238 2        f2 = source.fcb(2) and 80h;
239 2        f3 = source.fcb(3) and 80h;
240 2        f4 = source.fcb(4) and 80h;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 570

PACIFIC GROVE, CA 93950

SER. # _____

```

242 2      ra = source.fcb(9) and 80h;
243 2      sys = source.fcb(10) and 80h;
244 2      dcnt = reftsize(.source);
245 2      call move(.source.fcb(33),.filesize,3);
246 2      SOURCE.FCB(32) = 0;
247 2      source.fcb(33),source.fcb(34),source.fcb(35) = 0;
      /* cause immediate read with no preceding write */
248 2      NSOURCE = 0ffffh;
249 2      END SETUPSOURCE;

250 1      WRITEDEST: PROCEDURE;
      /* WRITE OUTPUT BUFFERS UP TO BUT NOT INCLUDING POSITION
      NDEST - THE LOW ORDER 7 BITS OF NDEST ARE ZERO */
251 2      DECLARE J BYTE,
      DATAOK BYTE,
      (tdest,n) address;
252 2      if not made then call setupdest;
254 2      if (n := ndest and 0ff80h) = 0 then return;
256 2      tdest = 0;
257 2      call setduser; /* destination user */
258 2      CALL SETDMA(.dbuf(tdest));
259 2      if (sparfil := (sparfil or insparc)) then
      /* set up fcb from random record no. */
260 2          do; if write$random(.dest) = 255 then
262 3              call error(1,.dest); /* DISK WRITE ERROR */
263 3          end;
      else
264 2          CALL SETRANDOM(.DEST); /* SET BASE RECORD FOR VERIFY */

265 2          do while tdest < n;
      /* SET DMA ADDRESS TO NEXT BUFFER */
266 3          CALL SETDMA(.dbuf(tdest));
267 3          call diskwrite(.dest);
268 3          IF dcnt > 0 THEN
269 3              call error(1,.dest); /* DISK WRITE ERROR */
270 3          tdest = tdest + 128;
271 3          END;
272 2      IF VERIF THEN /* VERIFY DATA WRITTEN OK */
273 2          DO;
274 3              tdest = 0;
275 3              CALL SETDMA(.BUFF); /* FOR COMPARE */
276 3              do while tdest < n;
277 4                  DATAOK = (RDRANDOM(.DEST) = 0);
278 4                  DESTR = DESTR + 1; /* NEXT RANDOM READ */
279 4                  J = 0;
      /* PERFORM COMPARISON */
280 4                  DO WHILE DATAOK AND J < 80h;
281 5                      DATAOK = (BUFF(J) = DBUFF(tdest+J));
282 5                      J = J + 1;
283 5                  END;
284 4                  tdest = tdest + 128;
285 4                  IF NOT DATAOK THEN
286 4                      call error(2,.dest); /* VERIFY ERROR */
287 4                  END;
288 3              call diskwrite(.dest);
      /* NOW READY TO CONTINUE THE WRITE OPERATION */
289 3          END;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

290 2      call move(.dbuff(tdest),.dbuff(0),low(ndest := ndest - tdest));
291 2      END WRITEDEST;

292 1      FILLSOURCE: PROCEDURE;
          /* FILL THE SOURCE BUFFER */
293 2      CALL SETUSER; /* SOURCE USER NUMBER SET */
294 2      nsource = nsbuf;
295 2      do while nsbuf < sblen;
          /* SET DMA ADDRESS TO NEXT BUFFER POSITION */
296 3      CALL SETDMA(.SBUF(nsbuf));
297 3      extsave = source.fcb(12); /* save extent field */
298 3      call diskrd(.source);
299 3      IF dcnt <> 0 THEN
300 3          DO; IF dcnt < 1 THEN -
302 4              call error(0,.source); /* DISK READ ERROR */
          /* END - OF - FILE */
          /* check boundry condition for bug in bdos and correct */
303 4          if (source.fcb(12) <> extsave) and (source.fcb(32) = 80h) then
304 4              source.fcb(32) = 0; /* zero current record */
305 4          call set$random(.source);
306 4          if (insparc := not maxsize) and (concat or not fastcopy) then
307 4              call error(18,.source); /* invalid format with sparse file */
308 4          endofsrc = true; /* set end of source file */
309 4          SBUF(nsbuf) = ENDFILE; return;
311 4          END;
          ELSE
312 3              nsbuf = nsbuf + 128;
313 3          END;
314 2      END FILLSOURCE;

315 1      PUTCHAR: PROCEDURE(B);
316 2      DECLARE B BYTE;
          /* WRITE BYTE B TO THE DESTINATION DEVICE GIVEN BY ODEST.TYPE */
317 2      IF B >= ' ' THEN
318 2          DO; COLUMN = COLUMN + 1;
320 3          IF DELET > 0 THEN /* MAY BE PAST RIGHT SIDE */
321 3              DO; IF COLUMN > DELET THEN RETURN;
324 4              END;
325 3          END;
326 2      if echo then call mon1(2,b); /* echo to console */
328 2      do case odest.type;
          /* CASE 0 IS OUT */
329 3          CALL OUTD(B);
          /* CASE 1 IS PRN, TABS EXPANDED, LINES LISTED */
330 3          call mon1(5,b);
          /* CASE 2 IS LST */
331 3          CALL MON1(5,B);
          /* CASE 3 IS axo */
332 3          CALL MON1(4,B);
          /* CASE 4 IS DESTINATION FILE */
333 3          DO;
334 4              IF NDEST >= DBLEN THEN CALL WRITEDEST;
336 4              DBUFF(NDEST) = B;
337 4              NDEST = NDEST+1;
338 4              END;
          /* CASE 5 IS CON */
339 3          CALL MON1(2,B);

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

340 3      END; /* of case */
341 2      END PUTDCHAR;

342 1      PUTDESTC: PROCEDURE(B);
343 2      DECLARE (B,I) BYTE;
          /* WRITE DESTINATION CHARACTER, TAB EXPANSION */
344 2      IF B <> TAB THEN CALL PUTDCHAR(B);
346 2      ELSE IF TABS = 0 THEN CALL PUTDCHAR(B);
          ELSE /* B IS TAB CHAR, TABS > 0 */
348 2          DO; I = COLUMN;
350 3          DO WHILE I >= TABS;
351 4          I = I - TABS;
352 4          END;
353 3          I = TABS - I;
354 3          DO WHILE I > 0;
355 4          I = I - 1;
356 4          CALL PUTDCHAR(' ');
357 4          END;
358 3      END;
359 2      IF B = CR THEN COLUMN = 0;
361 2      END PUTDESTC;

362 1      PRINT1: PROCEDURE(B);
363 2      DECLARE B BYTE;
364 2      IF (ZEROSUP := ZEROSUP AND B = 0) THEN
365 2          CALL PUTDESTC(' ');
          ELSE
366 2          CALL PUTDESTC('0'+B);
367 2      END PRINT1;

368 1      PRINTDIG: PROCEDURE(D);
369 2      DECLARE D BYTE;
370 2      CALL PRINT1(SHR(D,4)); CALL PRINT1(D AND 1111B);
372 2      END PRINTDIG;

373 1      NEWLINE: PROCEDURE;
374 2      DECLARE ONE BYTE;
375 2      ONE = 1;
376 2      ZEROSUP = (NUMB = 1);
377 2      C1 = DEC(C1+ONE); C2 = DEC(C2 PLUS 0); C3 = DEC(C3 PLUS 0);
380 2      CALL PRINTDIG(C3); CALL PRINTDIG(C2); CALL PRINTDIG(C1);
383 2      IF NUMB = 1 THEN /* USUALLY PRINTER OUTPUT */
384 2          DO; CALL PUTDESTC(';'); CALL PUTDESTC(' ');
387 3      END;
          ELSE
388 2          CALL PUTDESTC(TAB);
389 2      END NEWLINE;

390 1      PUTDEST: PROCEDURE(B);
391 2      DECLARE (I,B) BYTE;
          /* WRITE DESTINATION CHARACTER, CHECK TABS AND LINES */
392 2      IF FORMF THEN /* SKIP FORM FEEDS */
393 2          DO; IF B = FF THEN RETURN;
394 3      END;
397 2      IF PUTNUM THEN /* END OF LINE OR START OF FILE */
398 2          DO;
399 3          IF B <> FF THEN /* NOT FORM FEED */

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

400 3      DO;
401 4      IF (I:=PAGCNT) < 0 THEN /* PAGE EJECT */
402 4          DO; IF I=1 THEN I=LPP;
405 5          IF (LINENO := LINENO + 1) >= I THEN
406 5              DO; LINENO = 0; /* NEW PAGE */
408 6              CALL PUTDESTC(FF);
409 6              END;
410 5          END;
411 4      IF NUMB > 0 THEN
412 4          CALL NEWLINE;
413 4      PUTNUM = FALSE;
414 4      END;
415 3      END;
416 2      IF B = FF THEN LINENO = 0;
418 2      CALL PUTDESTC(B);
419 2      IF B = LF THEN PUTNUM = TRUE;
421 2      END PUTDEST;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

422 1      UTRAN: PROCEDURE(B) BYTE;
423 2          DECLARE B BYTE;
          /* TRANSLATE ALPHA TO UPPER CASE */
424 2          IF B >= 110$0001B AND B <= 111$1010B THEN /* LOWER CASE */
425 2              B = B AND 101$1111B; /* TO UPPER CASE */
426 2          RETURN B;
427 2          END UTRAN;

```

```

428 1      LTRAN: PROCEDURE(B) BYTE;
429 2          DECLARE B BYTE;
          /* TRANSLATE TO LOWER CASE ALPHA */
430 2          IF B >= 'A' AND B <= 'Z' THEN
431 2              B = B OR 10$0000B; /* TO LOWER */
432 2          RETURN B;
433 2          END LTRAN;

```

```

434 1      GETSOURCEC: PROCEDURE BYTE;
          /* READ NEXT SOURCE CHARACTER */
435 2          DECLARE (B,CONCHK) BYTE;

436 2          CONCHK = TRUE; /* CONSOLE STATUS CHECK BELOW */
437 2          DO CASE source.type;
          /* CASE 0 IS out */
438 3              go to notsource;
          /* CASE 1 IS prn */
439 3              go to notsource;
          /* CASE 2 IS 1st */
440 3              notsource;
                  call error(4,0); /* INVALID SOURCE */
          /* CASE 3 IS axo */
441 3              go to notsource;
          /* CASE 4 IS SOURCE FILE */
442 3              DO; IF NSOURCE >= SBLEN THEN
444 4                  do; if dblbuf then
446 5                      nsbuf = 0;
447 5                  else if nsource < 0ffffh then
448 5                      do; call writedest;
450 6                      nsbuf = ndest;

```

```

451 6          end;
          CALL FILLSOURCE;
453 5          end;
454 4          B = SBUFF(NSOURCE);
455 4          NSOURCE = NSOURCE + 1;
456 4          END;
          /* CASE 5 IS CON */
457 3          DO; CONCHK = FALSE; /* DON'T CHECK CONSOLE STATUS */
459 4          B = MON2(1,0);
460 4          END;
          /* CASE 6 IS oxi */
461 3          B = MON2(3,0) AND 7FH;
          /* CASE 7 IS INP */
462 3          B = INPD;
463 3          END; /* OF CASES */

464 2          IF CONCHK THEN /* TEST FOR CONSOLE CHAR READY */
465 2              DO;
466 3                  IF obj THEN /* SOURCE IS AN OBJECT FILE */
467 3                      CONCHK = ((CONCNT := CONCNT + 1) = 0);
                      ELSE /* ASCII */
468 3                          CONCHK = (B = LF);
469 3                  IF CONCHK THEN
470 3                      DO; IF CONBRK THEN
472 4                          DO;
473 5                          IF RDCHAR = ENDFILE THEN RETURN ENDFILE;
475 5                          call error(5,0); /* USER ABORTED */
476 5                          END;
477 4                      END;
478 3                      END;
479 2                      IF ZEROP THEN B = B AND 7FH;
481 2                      IF UPPER THEN RETURN UTRAN(B);
483 2                      IF LOWER THEN RETURN LTRAN(B);
485 2                      RETURN B;
486 2                      END GETSOURCEC;

497 1          GETSOURCE: PROCEDURE BYTE;
          /* GET NEXT SOURCE CHARACTER */
488 2          DECLARE CHAR BYTE;
489 2          MATCH: PROCEDURE(B) BYTE;
          /* MATCH START AND QUIT STRINGS */
490 3          DECLARE (B,C) BYTE;
491 3          IF (C:=COMBUFF(B:=(B+MATCHLEN))) = ENDFILE THEN /* END MATCH */
492 3              DO; COMBUFF(B) = CHAR; /* SAVE CURRENT CHARACTER */
494 4              RETURN TRUE;
495 4              END;
496 3              IF C = CHAR THEN MATCHLEN = MATCHLEN + 1;
          ELSE
498 3              MATCHLEN = 0; /* NO MATCH */
499 3              RETURN FALSE;
500 3              END MATCH;

501 2          IF QUITLEN > 0 THEN
502 2              DO; IF (QUITLEN := QUITLEN - 1) = 1 THEN RETURN LF;
505 3              RETURN ENDFILE; /* TERMINATED WITH CR,LF,ENDFILE */
506 3              END;
507 2          DO FOREVER; /* LOOKING FOR START */

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

508 3      IF FEEDLEN > 0 THEN /* GET SEARCH CHARACTERS */
509 3          DO; FEEDLEN = FEEDLEN - 1;
511 4          CHAR = COMBUFF(FEEDBASE);
512 4          FEEDBASE = FEEDBASE + 1;
513 4          RETURN CHAR;
514 4          END;
515 3      IF (CHAR := GETSOURCEC) = ENDFILE THEN RETURN ENDFILE;
517 3      IF STARTS > 0 THEN /* LOOKING FOR START STRING */
518 3          DO; IF MATCH(STARTS) THEN
520 4              DO; FEEDBASE = STARTS; STARTS = 0;
523 5              FEEDLEN = MATCHLEN + 1;
524 5              matchlen = 0;
525 5              END; /* OTHERWISE NO MATCH, SKIP CHARACTER */
526 4              END;
527 3      ELSE IF QUILTS > 0 THEN /* PASS CHARACTERS TIL MATCH */
528 3          DO; IF MATCH(QUILTS) THEN
530 4              DO; QUILTS = 0; QUITLEN = 2;
                    /* SUBSEQUENTLY RETURN CR, LF, ENDFILE */
533 5              RETURN CR;
534 5              END;
535 4              RETURN CHAR;
536 4              END;
537 3      ELSE
538 3          RETURN CHAR;
539 2      END; /* OF DO FOREVER */
540 2      END GETSOURCE;

540 1      RD$EOF: PROCEDURE BYTE;
541 2          /* RETURN TRUE IF END OF FILE */
542 2          CHAR = GETSOURCE;
543 2          IF obj THEN RETURN (endofsrc and (nsource > nsbuf));
544 2          RETURN (CHAR = ENDFILE);
545 2          END RD$EOF;

546 1      HEXRECORD: PROCEDURE;
547 2          DECLARE (h, hbuf, RL, CS, RT) BYTE,
                    zerorec byte, /* true if last record had length of zero */
                    LDA ADDRESS; /* LOAD ADDRESS WHICH FOLLOWS : */

548 2          ckhex: procedure byte;
549 3              IF H - '0' <= 9 THEN
550 3                  RETURN H-'0';
551 3              IF H - 'A' > 5 THEN
552 3                  CALL error(13,source); /* invalid hex digit */
553 3                  RETURN H - 'A' + 10;
554 3              end ckhex;

555 2          rdhex: procedure byte;
556 3              call putdest(h := getsources);
557 3              return ckhex;
558 3              end rdhex;

559 2      RD$CS: PROCEDURE BYTE;
560 3          /* READ BYTE WITH CHECKSUM */
561 3          RETURN CS := CS + (SHL(RDHEX,4) OR RDHEX);
562 3          END RD$CS;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

562 2      RDADDR: PROCEDURE ADDRESS;
          /* READ DOUBLE BYTE WITH CHECKSUM */
563 3      RETURN SHL(DOUBLE(RDCS),8) OR RDCS;
564 3      END RDADDR;

/* READ HEX FILE AND CHECK EACH RECORD
FOR VALID DIGITS, AND PROPER CHECKSUM */
565 2      zerorec = false;
          /* READ NEXT RECORD */
566 2      h = getsource;
567 2      do forever;
          /* SCAN FOR THE ':' */
568 3          DO WHILE h < ':';
569 4          IF (h = ENDFILE) THEN
570 4              do; if zerorec then return;
573 5              CALL error(15,.source); /* unexpected end of hex file */
574 5              end;
575 4              call putdest(h);
576 4              h = getsource;
577 4              END;

          /* ':' FOUND */
          /* check for end of hex record */
578 3      h = getsource;
579 3      rl = shl(ckhex,4);
580 3      hbuf = h; h = getsource;
582 3      rl = rl or ckhex;
583 3      if (rl = 0) then zerorec = true;
585 3      else zerorec = false;
586 3      if (zerorec and ignor) then
587 3          do while (h < ':') and (h < endfile);
588 4          h = getsource;
589 4          end;
590 3      else do; call putdest(':');
592 4          call putdest(hbuf);
593 4          call putdest(h);
594 4          cs = rl;
595 4          LDA = RDADDR; /* LOAD ADDRESS */

          /* READ WORDS UNTIL RECORD LENGTH EXHAUSTED */
596 4      RT = RDCS; /* RECORD TYPE */
597 4      DO WHILE RL < 0; RL = RL - 1;
599 5      hbuf = RDCS;
          /* INCREMENT LA HERE FOR EXACT ADDRESS */
600 5      END;

          /* CHECK SUM */
601 4      IF rdcS < 0 THEN
602 4          CALL error(9,.source); /* hex record checksum */
603 4          h = getsource;
604 4          end;
605 3      end; /* do forever */
606 2      END HEXRECORD;

607 1      CK$STRINGS: PROCEDURE;
608 2      IF STARTS > 0 THEN

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____


```

609 2      call error(11,0); /* START NOT FOUND */
610 2      IF QUIFS > 0 THEN
611 2          call error(12,0); /* QUIT NOT FOUND */
612 2      END CK$STRINGS;

613 1      CLOSEDEST: PROCEDURE;
614 2          DO WHILE (LOW(NDEST) AND 7FH) <> 0;
615 3          CALL PUTDEST(ENDFILE);
616 3          END;
617 2      CALL CK$STRINGS;
618 2      CALL WRITEDEST;
619 2      call setduser; /* destination user */
620 2      CALL CLOSE(.DEST);
621 2      IF DCNT = 255 THEN
622 2          call error(14,.dest); /* CLOSE FILE */
623 2      call open(.odest);
624 2      IF DCNT <> 255 THEN /* FILE EXISTS */
625 2          DO;
626 3          call close(.odest);
627 3          IF ROL(odest.fcb(9),1) THEN /* READ ONLY */
628 3              DO;
629 4              IF NOT WRROF THEN
630 4                  DO;
631 5                  do while ((dcnt <> 'Y') and (dcnt <> 'N'));
632 6                  CALL PRINT(,('DESTINATION IS R/O, DELETE (Y/N)?'));
633 6                  dcnt = utran(rdchar);
634 6                  end;
635 5                  IF dcnt <> 'Y' THEN
636 5                      DO; CALL PRINT(,('**NOT DELETED**'));
637 5                      CALL CRLF;
638 6                      CALL DELETE(.DEST);
639 6                      RETURN;
640 6                      END;
641 6                  CALL CRLF;
642 5                  END;
643 5                  END;
644 4          END;

/* reset r/o and sys attributes */
645 3      odest.fcb(9) = odest.fcb(9) and 7fh;
646 3      odest.fcb(10) = odest.fcb(10) AND 7FH;
647 3      CALL SETIND(.odest);
648 3      CALL DELETE(.odest);
649 3      END;

650 2      CALL MOVE(.odest.fcb,.dest.fcb(16),16); /* READY FOR RENAME */
651 2      CALL RENAME(.DEST);

/* set destination attributes same as source */
652 2      odest.fcb(1) = (odest.fcb(1) and 07fh) or f1;
653 2      odest.fcb(2) = (odest.fcb(2) and 07fh) or f2;
654 2      odest.fcb(3) = (odest.fcb(3) and 07fh) or f3;
655 2      odest.fcb(4) = (odest.fcb(4) and 07fh) or f4;
656 2      odest.fcb(8) = (odest.fcb(8) and 07fh);
657 2      odest.fcb(9) = (odest.fcb(9) and 07fh) or ra;
658 2      odest.fcb(10) = (odest.fcb(10) and 07fh) or sys;
659 2      odest.fcb(11) = (odest.fcb(11) and 07fh);
660 2      call setind(.odest);
661 2      if archiv then /* set archive bit */
662 2          do; call setsuser;
663 2          source.fcb(11) = source.fcb(11) or 080h;

```

CP/M-86 v.1.1 (vol. II)

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # C81-162

```

665 3      source.fcb(12) = 0;
666 3      call setind(.source);
667 3      end;
668 2      END CLOSEDEST;

669 1      SIZE$MEMORY: PROCEDURE;
        /* SET UP SOURCE AND DESTINATION BUFFERS */
670 2      SBASE = .MEMORY + SHR(MAXB - .MEMORY,1);
671 2      SBLN, DBLEN = SHR((MAXB - .MEMORY) AND OFF00H,1) - 128;
672 2      if not dblbuf then
673 2          do; /* ABSORB THE SOURCE BUFFER INTO THE DEST BUFFER */
674 3              SBASE = .MEMORY;
675 3              IF DBLEN >= 4000H THEN DBLEN,sblen = 7F80H;
677 3              ELSE DBLEN,sblen = DBLEN + SBLN;
678 3              end;
679 2      else do; /* may need to write destination buffer */
680 3          if ndest >= dblen then call writedest;
682 3          nsbuf = 0;
683 3          end;
684 2      END SIZE$MEMORY;

685 1      setupeob: procedure;
        /* sets nsbuf to end of source buffer */
686 2      declare i byte;
687 2      if not obj then
688 2          do; tblen = nsbuf - 128;
690 3          do i = 0 to 128;
691 4              if (sbuff(tblen + i)) = endfile then
692 4                  do; nsbuf = tblen + i;
694 5                      return;
695 5                  end;
696 4          end;
697 3      end;
698 2      end setupeob;

699 1      SIMPLCOPY: PROCEDURE;
700 2      DECLARE I BYTE;
701 2      declare
        fast lit '0', /* fast file to file copy */
        chrt lit '1', /* character transfer option */
        dubl lit '2'; /* double buffer required for file copy */
702 2      declare optype(26) byte data (
        /* option type for each option character */
        fast, /* for A option */
        fast, /* for B option */
        fast, /* for C option */
        dubl, /* for D option */
        chrt, /* for E option */
        dubl, /* for F option */
        fast, /* for G option */
        chrt, /* for H option */
        dubl, /* for I option */
        fast, /* for J option */
        fast, /* for K option */
        chrt, /* for L option */
        fast, /* for M option */
        dubl, /* for N option */

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. B. X 579

PACIFIC GROVE, CA 93950

SER. # _____

```

fast, /* for O option */
dubl, /* for P option */
dubl, /* for Q option */
fast, /* for R option */
dubl, /* for S option */
dubl, /* for T option */
chrt, /* for U option */
fast, /* for V option */
fast, /* for W option */
fast, /* for X option */
fast, /* for Y option */
chrt); /* for Z option */

```

```

703 2      chkrandom: procedure;
704 3          call setsuser;
705 3          call set$random(.source);
706 3          call setdma(.buff);
707 3          do forever;
708 4              if (dcnt := rd$random(.source)) = 0 then
709 4                  do; destr = sourcer;
711 5                      destr2 = sourcer2;
712 5                      endofsrc = false;
713 5                      return;
714 5                      end;
715 4              if dcnt = 1 then
716 4                  do; if (sourcer := sourcer + 1) = 0 then
718 5                      sourcer2 = sourcer2 + 1;
719 5                      end;
720 4              else if dcnt = 4 then
721 4                  do; if (sourcer := (sourcer + 128) and 0ff80h) = 0 then
723 5                      sourcer2 = sourcer2 + 1;
724 5                      end;
725 4              else call error(0,.source);
726 4              end;
727 3          end chkrandom;

728 2      fastcopy = (sfile and dfile);
729 2      endofsrc = false;
730 2      dblbuf = false;
731 2      sparfil = false;
732 2      /* LOOK FOR PARAMETERS */
733 3      DO I = 0 TO 25;
734 3      IF CONT(I) <> 0 THEN
735 4          DO;
736 4          IF optype(i) = chrt THEN
737 4              FASTCOPY = FALSE;
738 4          else
739 4              if optype(i) = dubl then
740 5                  do; dblbuf = (sfile and dfile);
741 5                  fastcopy = false;
742 5                  end;
743 3          END;
744 3          END;

744 2      CALL SIZE$MEMORY;
745 2      if sfile then
746 2          CALL SETUP$SOURCE;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```
/* FILES READY FOR COPY */
```

```

747 2      if fastcopy then
748 2          do while not endofsrc;
749 3          CALL FILLSOURCE;
750 3          if (endofsrc and not insparc) then
751 3              do; call setsuser;
753 4              call close(.source);
754 4              if concat then
755 4                  do; call setupeob;
757 5                  ndest = nsbuf;
758 5                  if nendcmd then return;
760 5                  end;
761 4              end;
762 3          ndest = nsbuf;
763 3          CALL WRITEDEST;
764 3          nsbuf = ndest;
765 3          if (endofsrc and insparc) then
766 3              call chkrandom;
767 3          end;

768 2      else do;
769 3          /* PERFORM THE ACTUAL COPY FUNCTION */
770 3          IF HEXT OR IGNOR THEN /* HEX FILE */
771 3              call hexrecord;
772 3          ELSE
773 3              DO WHILE NOT RD$EOF;
774 3              CALL PUTDEST(CHAR);
775 3              END;
776 3          if concat and nendcmd then
777 3              do; nsbuf = ndest;
778 3              return;
779 3              end;
780 2          if dfile then
781 2              CALL CLOSEDEST;
782 2          END SIMPLECOPY;

783 1      MULTICOPY: PROCEDURE;
784 2          DECLARE (NEXTDIR, NDCNT, NCOPIED) ADDRESS;

785 2          PRNAME: PROCEDURE;
786 3              /* PRINT CURRENT FILE NAME */
787 3              DECLARE (I,C) BYTE;
788 3              CALL CRLF;
789 3              DO I = 1 TO FNSIZE;
790 3              IF (C := odest.fcb(I)) <> ' ' THEN
791 3                  DO; IF I = FEXT THEN CALL PRINTCHAR(' ');
792 3                  CALL PRINTCHAR(C);
793 3                  END;
794 3              END;
795 3          END;
796 3          END PRNAME;

797 2      archck: procedure byte;
798 3          /* check if archive bit is set in any extent of source file */
799 3          call setsuser;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

799 3      source.fcb(12) = what;
800 3      call search(.source);
801 3      do while dcnt <> 255;
802 4      call move(.buff+shl(dcnt and 11b,5)+1,.source.fcb(1),15);
803 4      if not rol(source.fcb(11),1) then
804 4          do; source.fcb(12) = 0;
806 5          return 1;
807 5          end;
808 4      call searchn;
809 4      end;
810 3      return 0;
811 3      end archck;

/* initialize counters */
812 2      NEXTDIR, NCOPIED = 0;

813 2      DO FOREVER;
/* FIND A MATCHING ENTRY */
814 3      CALL SETUSER; /* SOURCE USER */
815 3      CALL SETDMA(.BUFF);
816 3      searfc(12) = 0;
817 3      CALL SEARCH(.SEARFCB);
818 3      NDCNT = 0;
819 3      DO WHILE (DCNT <> 255) AND NDCNT < NEXTDIR;
820 4          NDCNT = NDCNT + 1;
821 4          CALL SEARCHN;
822 4          END;
/* FILE CONTROL BLOCK IN BUFFER */
823 3      IF DCNT = 255 THEN
824 3          DO; IF NCOPIED = 0 THEN
826 4              call error(10,.searfc); /* file not found */
827 4              if not kilds then
828 4                  CALL CRLF;
829 4              RETURN;
830 4              END;
831 3          NEXTDIR = NDCNT + 1;
/* GET THE FILE CONTROL BLOCK NAME TO DEST */
832 3          CALL MOVE(.BUFF + SHL(DCNT AND 11B,5)+1,.odest.fcb(1),15);
833 3          CALL MOVE(.odest.fcb(1),.SOURCE.FCB(1),15); /* FILL BOTH FCB'S */
834 3          if not archiv or archck then
835 3              do; odest.fcb(12) = 0;
837 4              IF RSYS OR NOT ROL(odest.fcb(10),1) THEN /* OK TO READ */
838 4                  DO; if not kilds then /* kill display option */
840 5                      do; IF NCOPIED = 0 THEN
842 6                          CALL PRINT(.(('COPYING -$')));
843 6                          CALL PRNAME;
844 6                          end;
845 5                      ncopied = ncopied + 1;
846 5                      made = false; /* destination file not made */
847 5                      CALL SIMPLECOPY;
848 5                      END;
849 4                  end;
850 3          END;
851 2      END MULTCOPY;

352 1      CK$DISK: PROCEDURE;
/* error if same user and same disk */

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

853 2      IF (odest.user = source.user) and (odest.fcb(0) = source.fcb(0)) THEN
854 2          CALL FORMERR;
855 2      END CK$DISK;

```

```

856 1      GNC: PROCEDURE BYTE;
857 2          IF (CBP := CBP + 1) >= COMLEN THEN RETURN CR;
859 2          RETURN UTRAN(CONBUFF(CBP));
860 2      END GNC;

```

```

861 1      DEBLANK: PROCEDURE;
862 2          DO WHILE (CHAR := GNC) = ' ';
863 3              END;
864 2      END DEBLANK;

```

```

865 1      CK$EOL: PROCEDURE;
866 2          CALL DEBLANK;
867 2          IF CHAR <> CR THEN CALL FORMERR;
868 2      END CK$EOL;

```

```

870 1      SCAN: PROCEDURE(FCBA);
871 2          DECLARE FCBA ADDRESS,          /* ADDRESS OF FCB TO FILL */
          fcb based fcba structure (      /* FCB STRUCTURE */
              fcb(fsize) byte,
              user byte,
              type byte );
872 2          DECLARE (I,K) BYTE; /* TEMP COUNTERS */

```

/* SCAN LOOKS FOR THE NEXT DELIMITER, DEVICE NAME, OR FILE NAME.
THE VALUE OF CBP MUST BE 255 UPON ENTRY THE FIRST TIME */

```

873 2      DELIMITER: PROCEDURE(C) BYTE;
874 3          DECLARE (I,C) BYTE;
875 3          DECLARE DEL(*) BYTE DATA
          (' =,.;, <> ', CR, LA, LB, RB);
876 3          DO I = 0 TO LAST(DEL);
877 4              IF C = DEL(I) THEN RETURN TRUE;
879 4          END;
880 3          RETURN FALSE;
881 3      END DELIMITER;

```

```

882 2      PUTCHAR: PROCEDURE;
883 3          FCBS.FCB(FLEN:=FLEN+1) = CHAR;
884 3          IF CHAR = WHAT THEN AMBIG = TRUE; /* CONTAINS AMBIGUOUS REF */
886 3      END PUTCHAR;

```

```

887 2      FILLQ: PROCEDURE(LEN);
          /* FILL CURRENT NAME OR TYPE WITH QUESTION MARKS */
888 3          DECLARE LEN BYTE;
889 3          CHAR = WHAT; /* QUESTION MARK */
890 3          DO WHILE FLEN < LEN;
891 4              CALL PUTCHAR;
892 4          END;
893 3      END FILLQ;

```

```

894 2      SCANPAR: PROCEDURE;
895 3          DECLARE (I,J) BYTE;
          /* SCAN OPTIONAL PARAMETERS */

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

076 3      CHAR = GNC; /* SCAN PAST BRACKET */
097 3      DO WHILE NOT(CHAR = CR OR CHAR = RB);
098 4      IF (I := CHAR - 'A') > 25 THEN /* NOT ALPHA */
099 4          DO; IF CHAR = ' ' THEN
101 5              CHAR = GNC;
              ELSE
102 5                  call error(6,0); /* BAD PARAMETER */
103 5              END;
              ELSE
104 4                  DO; /* SCAN PARAMETER VALUE */
105 5                      IF CHAR = 'S' OR CHAR = 'Q' THEN
106 5                          DO; /* START OR QUIT COMMAND */
107 6                              J = CBP + 1; /* START OF STRING */
108 6                              DO WHILE NOT ((CHAR := GNC) = ENDFILE OR CHAR = CR);
109 7                                  END;
110 6                                  CHAR=GNC;
111 6                                  END;
112 5                      ELSE IF (J := (CHAR := GNC) - '0') > 9 THEN
113 5                          J = 1;
                          ELSE
114 5                              DO WHILE (K := (CHAR := GNC) - '0') <= 9;
115 6                                  J = J * 10 + K;
116 6                                  END;
117 5                              CONT(I) = J;
118 5                              IF I = 6 THEN /* SET SOURCE USER */
119 5                                  DO;
120 6                                      IF J > 15 THEN
121 6                                          call error(7,0); /* INVALID USER NUMBER */
122 6                                          fcbs.user = J;
123 6                                          END;
124 5                                  END;
125 4                              END;
126 3                          CHAR = GNC;
127 3                      END SCANPAR;

```

/* scan procedure entry point */

```

/* INITIALIZE FILE CONTROL BLOCK TO EMPTY */
128 2  fcbs.type = ERR; CHAR = ' '; FLEN = 0;
131 2      DO WHILE FLEN < FRSIZE -1;
132 3      IF FLEN = FNSIZE THEN CHAR = 0;
134 3      CALL PUTCHAR;
135 3      END;
136 2  fcbs.fcb(0) = cdisk +1; /* initialize to current disk */
137 2  fcbs.user = cuser; /* and current user */
/* CLEAR PARAMETERS */
138 2      DO I = 0 TO 25; CONT(I) = 0;
140 3      END;
141 2  FEEDLEN, MATCHLEN, QUITLEN = 0;

/* DEBLANK COMMAND BUFFER */
142 2  CALL DEBLANK;

/* CHECK PERIPHERALS AND DISK FILES */
/* SCAN NEXT NAME */
143 2  DO FOREVER;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

944 3      FLEN = 0;
945 3      DO WHILE NOT DELIMITER(CHAR);
946 4      IF FLEN >= NSIZE THEN /* ERROR, FILE NAME TOO LONG */
947 4          RETURN;
948 4      IF CHAR = '*' THEN CALL FILLQ(NSIZE);
950 4      ELSE CALL PUTCHAR;
951 4      CHAR = GNC;
952 4      END;

/* CHECK FOR DISK NAME OR DEVICE NAME */
953 3      IF CHAR = ':' THEN
954 3          DO; IF FLEN = 1 THEN
955 3              /* MAY BE DISK NAME A ... P */
956 4              DO;
957 5              IF (fcbs.fcb(0) := fcbs.fcb(1) - 'A' + 1) > 16 THEN
958 5                  RETURN; /* ERROR, INVALID DISK NAME */
959 5              CALL DEBLANK; /* MAY BE DISK NAME ONLY */
960 5              IF DELIMITER(CHAR) THEN
961 5                  DO; IF CHAR = LB THEN
962 6                      CALL SCANPAR;
963 6                      CBP = CBP - 1;
964 6                      fcbs.type = DISKNAME;
965 6                      RETURN;
966 6                      END;
967 6                  END;
968 5              END;
969 4          ELSE
970 4              /* MAY BE A THREE CHARACTER DEVICE NAME */
971 4              IF FLEN <> 3 THEN /* ERROR, CANNOT BE DEVICE NAME */
972 4                  RETURN;
973 5              ELSE
974 5                  /* LOOK FOR DEVICE NAME */
975 5                  DO; DECLARE (I,J,K) BYTE, N LITERALLY '9',
976 5                  ID(*) BYTE DATA
977 5                  ('OU|PRNLSTAXO',
978 5                  0,0,0, /* fake area for file type */
979 5                  'CONAXIINPNULOF',0);
980 5                  J = 255;
981 5                  DO K = 0 TO M;
982 5                      I = 0;
983 5                      DO WHILE ((I:=I+1) <= 3) AND
984 5                      ID(J+I) = fcbs.fcb(I);
985 5                          END;
986 5                      IF I = 4 THEN /* COMPLETE MATCH */
987 5                          DO; fcbs.type = k;
988 5                          /* SCAN PARAMETERS */
989 5                          IF GNC = LB THEN CALL SCANPAR;
990 5                          CBP = CBP - 1;
991 5                          RETURN;
992 5                          END;
993 5                      J = J + 3; /* OTHERWISE TRY NEXT DEVICE */
994 5                      END;
995 5                  RETURN; /* ERROR, NO DEVICE NAME MATCH */
996 5                  END;
997 4              IF CHAR = LB THEN /* PARAMETERS FOLLOW */
998 4                  CALL SCANPAR;
999 4              END;

```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____


```

ELSE
  /* CHAR IS NOT ':', SO FILE NAME IS SET. SCAN REMAINDER */
  DO; IF FLEN = 0 THEN /* ERROR, NO PRIMARY NAME */
    RETURN;
  FLEN = NSIZE;
  IF CHAR = '.' THEN /* SCAN FILE TYPE */
    DO WHILE NOT DELIMITER(CHAR := GNC);
    IF FLEN >= FNSIZE THEN
      RETURN; /* ERROR, TYPE FIELD TOO LONG */
    IF CHAR = '*' THEN CALL FILLQ(FNSIZE);
    ELSE CALL PUTCHAR;
  END;

  IF CHAR = LB THEN
    CALL SCANPAR;
  /* RESCAN DELIMITER NEXT TIME AROUND */
  CBP = CBP - 1;
  fcbs.type = FILE;
  FCBS.FCB(32) = 0;
  RETURN;
END;

END;
END SCAN;

1014 1  pla:
      /* PIP ENTRY POINT */
      /* BUFFER AT 30H CONTAINS REMAINDER OF LINE TYPED
      FOLLOWING THE COMMAND 'PIP' - IF ZERO THEN PROMPT TIL CR */
      CALL MOVE(.BUFF,.COMLEN,30H);
1015 1  MULTCOM = (COMLEN = 0);

      /* GET CURRENT CP/M VERSION */
      IF CVERSION < VERSION THEN
1016 1  DO;
1017 1  CALL PRINT(.(('REQUIRES CP/M-86$')));
1018 2  CALL BOOT;
1019 2  CALL BOOT;
1020 2  END;

1021 1  if cversion >= mversion then call mon1(45,255);

1023 1  if multcom then
1024 1  do; call printx(.(('CP/M-86 PIP VERSION 1.1$')));
1026 2  call crlf;
1027 2  end;

1028 1  CUSER = GETUSER; /* GET CURRENT USER */
1029 1  CDISK = getdisk; /* GET CURRENT DISK */

1030 1  RETRY:
      /* ENTER HERE ON ERROR EXIT FROM THE PROCEDURE 'ERROR' */
      /* MAIN PROCESSING LOOP. PROCESS UNTIL CR ONLY */
      DO FOREVER;
1031 2  C1, C2, C3 = 0; /* LINE COUNT = 000000 */
1032 2  CONCNT,COLUMN = 0; /* PRINTER TABS */
1033 2  ndest,nsbuf = 0;
1034 2  ambig = false;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. = _____

```

1035 2      made = false;      /* destination file not made */
1036 2      concat = false;
1037 2      PUTNUM = TRUE;      /* ACTS LIKE LF OCCURRED ON ASCII FILE */
1038 2      dfile,sfile = true;
1039 2      nendcmd = true;
1040 2      LINENO = 254;      /* INCREMENTED TO 255 > PAGCNT */
/* READ FROM CONSOLE IF NOT A ONELINER */
1041 2      IF MULTCOM THEN
1042 2          DO; CALL PRINTCHAR('*'); CALL RDCOM;
1045 3          CALL CRLF;
1046 3          END;
1047 2      CBP = 255;
1048 2      IF COMLEN <= 1 THEN /* single character or <CR> */
1049 2          do; call setcuser; /* restore current user */
1051 3          CALL BOOT; /* normal exit from pip here */
1052 3          end;

/* LOOK FOR SPECIAL CASES FIRST */
1053 2      CALL SCAN(.odest);
1054 2      if ambig then call error(3,.odest); /* invalid destination */
1056 2      call debblank; /* check for equal sign or left arrow */
1057 2      if (char <> '=') and (char <> la) then call formerr;
1057 2      call scan(.source);

1060 2      IF odest.type = DISKNAME THEN
1061 2          DO;
1062 3          IF source.type <> file then call formerr;
1064 3          CALL CK$EOL;
1065 3          CALL CK$DISK;
1066 3          odest.type = file; /* set for character transfer */
/* MAY BE MULTI COPY */
1067 3          IF AMBIG THEN /* FORM IS A:=B:AFN */
1068 3              DO;
1069 4              CALL MOVE(.source.fcb(0),.searfc(0),frsize);
1070 4              CALL MULTCOPY;
1071 4              END;
1072 3          ELSE DO; /* FORM IS A:=B:UFN */
1073 4              CALL MOVE(.source.fcb(1),.odest.fcb(1),frsize - 1);
1074 4              CALL SIMPCOPY;
1075 4              END;
1076 3          END;

1077 2      else IF (odest.type = FILE) and (source.type = DISKNAME) THEN
1078 2          DO;
1079 3          CALL CK$EOL;
1080 3          CALL CK$DISK;
1081 3          source.type = file; /* set for character transfer */
1082 3          call move(.odest.fcb(1),.source.fcb(1),(frsize - 1));
1083 3          CALL SIMPCOPY;
1084 3          END;

1085 2      else if (odest.type > cons) then
1086 2          call error(3,0); /* invalid destination */

1087 2      else do;
1088 3          IF odest.type <> FILE THEN dfile = false;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

/* SCAN AND COPY UNTIL CR */
1090 3      DO WHILE nendcmd;
1091 4      call deblank;
1092 4      IF (CHAR <> ',' AND CHAR <> CR) THEN
1093 4          call error(16,0); /* invalid separator */
1094 4      concat = concat or (nendcmd != (char = ','));
1095 4      IF odest.type = PRNT THEN
1096 4          DO; NUMB = 1;
1098 5          IF TABS = 0 THEN TABS = 8;
1100 5          IF PAGCNT = 0 THEN PAGCNT = 1;
1102 5          END;
1103 4      IF (source.type < file) or (source.type > eof) or ambig THEN
1104 4          call error(4,0); /* invalid source */
1105 4      IF source.type <> FILE THEN /* NOT A SOURCE FILE */
1106 4          sfile = false;
1107 4      IF source.type = MULT THEN
1108 4          /* SEND 40 NULLS TO OUTPUT DEVICE */
1109 4          DO sfile = 0 TO 39; CALL PUTDEST(0);
1110 5          END;
1111 4      ELSE IF source.type = EOFT THEN
1112 4          CALL PUTDEST(ENDFILE);
1113 4      else call simplecopy;

1114 4      CALL CK$STRINGS;
1115 4      /* READ ENDFILE, GO TO NEXT SOURCE */
1116 4      if nendcmd then call scan(.source);
1117 4      END;
1118 3      end;

/* COMLEN SET TO 0 IF NOT PROCESSING MULTIPLE COMMANDS */
1119 2      COMLEN = MULTCOM;

1120 2      END; /* DO FOREVER */
1121 1      END;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
47	0001H	2	A. WORD PARAMETER AUTOMATIC
66	0004H	2	A. WORD PARAMETER AUTOMATIC
27	0000H	2	A. WORD PARAMETER 28
30	0000H	2	A. WORD PARAMETER 31
24	0000H	2	A. WORD PARAMETER 25
157	0000H	1	A. BYTE BASED(S) 159
51	0001H	2	A. WORD PARAMETER AUTOMATIC
11	009CH	1	ANBIG. BYTE 885 1034 1054 1067 1103
797	12B4H	30	ARCHCK. PROCEDURE BYTE STACK=0012H
17	0135H	1	ARCHIV. BYTE AT 661 834
7			AXIT. LITERALLY
7			AXOT. LITERALLY
362	0004H	1	B. BYTE PARAMETER AUTOMATIC 363 364 366
157	0000H	1	B. BYTE BASED(D) 159
370	0004H	1	B. BYTE PARAMETER AUTOMATIC 391 394 399 416 418 419
19	0000H	1	B. BYTE PARAMETER 20
315	0004H	1	B. BYTE PARAMETER AUTOMATIC 316 317 327 329 330 331 332 336
342	0004H	1	B. BYTE PARAMETER AUTOMATIC 343 344 345 347 359
435	015AH	1	B. BYTE 454 459 461 462 468 480 482 484 485
489	0004H	1	B. BYTE PARAMETER AUTOMATIC 490 491 493
428	0004H	1	B. BYTE PARAMETER AUTOMATIC 429 430 431 432
422	0004H	1	B. BYTE PARAMETER AUTOMATIC 423 424 425 426
33	0268H	14	BOOT. PROCEDURE STACK=0008H 1019 1051
3	0000H	128	BUFF. BYTE ARRAY(128) EXTERNAL(2) 82 275 281 706 802 815 332
873	0004H	1	C. BYTE PARAMETER AUTOMATIC 874 877
490	015DH	1	C. BYTE 491 496
786	0167H	1	C. BYTE 789 793
18	0152H	1	C1. BYTE 377 382 1031
18	0151H	1	C2. BYTE 378 381 1031
18	0150H	1	C3. BYTE 379 380 1031
14	0132H	1	CBP. BYTE 857 859 907 964 983 1007 1047
13	00B0H	130	CEUFF. BYTE ARRAY(130) 13
10	0094H	1	CDISK. BYTE 936 1029
39	0004H	1	CHAR. BYTE PARAMETER AUTOMATIC 40 41
11	00A7H	1	CHAR. BYTE 541 544 772 862 867 883 884 889 896 897 898 900
488	015CH	1	CHAR. BYTE 901 905 908 910 912 914 926 929 933 945 948 951 953 960
703	1115H	107	CHKRANDOM. PROCEDURE STACK=0020H 766
701			CHRT. LITERALLY 702 735
852	1304H	26	CKDISK. PROCEDURE STACK=0024H 1065 1080
865	1352H	18	CKEOL. PROCEDURE STACK=0024H 1064 1079
543	008CH	42	CKHEX. PROCEDURE BYTE STACK=0020H 557 579 582
607	0DF5H	39	CKSTRINGS. PROCEDURE STACK=0020H 617 1114
73	0333H	53	CKUSER. PROCEDURE STACK=0008H 90 94
74	0320H	19	CLOSE. PROCEDURE STACK=000AH 189 620 626 753
613	0E1CH	329	CLOSEDEST. PROCEDURE STACK=004AH 781

COPYRIGHT 1981 1982

LORAIN RESEARCH

11200 X 879

PACIFIC GROVE, CA 93950

SER. =

52 54

834

363 364 366

391 394 399 416 418 419

316 317 327 329 330 331 332 336

343 344 345 347 359

468 480 482 484 485

490 491 493

429 430 431 432

423 424 425 426

1019 1051

82 275 281 706 802 815 332

1014

874 877

491 496

789 793

377 382 1031

378 381 1031

379 380 1031

857 859 907 964 983 1007 1047

13

936 1029

40 41

541 544 772 862 867 883 884 889 896 897 898 900

901 905 908 910 912 914 926 929 933 945 948 951 953 960

962 990 997 998 1001 1005 1057 1092 1094

493 496 511 513 515 535 537

766

702 735

1065 1080

1064 1079

557 579 582

617 1114

90 94

189 620 626 753

781

10	003EH	1	COLUMN	BYTE	319	322	349	360	1032										
13	00B2H	128	COMBUFF.	BYTE ARRAY(128) AT				491	493	511	859								
13	00B1H	1	COMLEN	BYTE AT	208	857	1014	1015	1048	1119									
60	02DFH	15	CONBRK	PROCEDURE BYTE STACK=0008H						471									
11	007BH	1	CONCAT	BYTE	306	754	774	1036	1094										
435	015BH	1	CONCHK	BYTE	436	458	464	467	468	469									
11	00ASH	1	CONCNT	BYTE	467	1032													
7			CONS	LITERALLY	1085														
17	013SH	26	CONT	BYTE ARRAY(26)		17	733	917	939										
6	0026H	31	COPYRIGHT.	BYTE ARRAY(31) DATA															
9			CR	LITERALLY	44	359	533	858	867	875	897	908	1092						
43	029AH	17	CRLF	PROCEDURE STACK=000EH				53	209	638	642	787	828	1026	1045				
547	0161H	1	CS	BYTE	560	594													
15	0133H	1	CUSER.	BYTE	131	937	1028												
63	02EEH	15	CVERSION	PROCEDURE WORD STACK=0008H						1016	1021								
155	0006H	2	D.	WORD PARAMETER AUTOMATIC				156	157	159	161								
368	0004H	1	D.	BYTE PARAMETER AUTOMATIC				369	370	371									
251	0156H	1	DATACK	BYTE	277	280	281	285											
11	009AH	1	DBLBUF	BYTE	445	672	730	739											
10	0002H	2	DBLEN.	WORD	334	671	675	676	677	680									
10	0000H	1024	DBUFF.	BYTE ARRAY(1024) AT				258	266	281	290	336							
12	00AFH	1	DCNT	BYTE	72	76	80	82	84	89	93	102	106	110	114	121			
					226	235	236	244	268	299	301	621	624	631	633	635	708	715	
					720	801	802	819	823	832									
861	1343H	15	DEBLANK.	PROCEDURE STACK=000CH				866	942	959	1056	1091							
			DEC.	BUILTIN	377	378	379												
875	019BH	12	DEL.	BYTE ARRAY(12) DATA				876	877										
17	013BH	1	DELET.	BYTE AT	320	322													
96	0393H	16	DELETE	PROCEDURE STACK=000AH				190	224	639	648								
873	1549H	46	DELIMITER.	PROCEDURE BYTE STACK=0004H				945	960	998									
10	002EH	38	DEST	STRUCTURE	10	189	190	222	223	224	225	227	228	261	262				
					264	267	269	277	286	288	620	622	639	650	651				
10	004FH	2	DESTR.	WORD AT	278	710													
10	0051H	1	DESTR2	BYTE AT	711														
11	007DH	1	DFILE.	BYTE	728	739	780	1038	1089										
7			DISKNAME	LITERALLY	965	1060	1077												
100	03A3H	19	DISKRD	PROCEDURE STACK=000AH				298											
104	03B6H	19	DISKWRITE.	PROCEDURE STACK=000AH				267	288										
			DOUBLE	BUILTIN	563														
701			DUBL	LITERALLY	702	737													
17	0137H	1	ECHO	BYTE AT	326														
4			ENDFILE.	LITERALLY	309	473	474	491	505	515	516	544	569	587	615				
					691	908	1112												
11	00A0H	1	ENDOFSRC	BYTE	308	543	712	729	748	750	765								
7			EQFT	LITERALLY	1103	1111													
166	0045H	10	ER00	BYTE ARRAY(10) DATA				185											
167	0047H	11	ER01	BYTE ARRAY(11) DATA				185											
168	005AH	7	ER02	BYTE ARRAY(7) DATA				185											
169	0061H	20	ER03	BYTE ARRAY(20) DATA				185											
170	0075H	15	ER04	BYTE ARRAY(15) DATA				185											
171	0084H	13	ER05	BYTE ARRAY(13) DATA				185											
172	0091H	14	ER06	BYTE ARRAY(14) DATA				185											
173	009FH	20	ER07	BYTE ARRAY(20) DATA				185											
174	00B3H	15	ER08	BYTE ARRAY(15) DATA				185											
175	00C2H	20	ER09	BYTE ARRAY(20) DATA				185											
176	00D6H	15	ER10	BYTE ARRAY(15) DATA				185											
177	00E5H	16	ER11	BYTE ARRAY(16) DATA				185											

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. B. X 570

PACIFIC GROVE, CA 93950

SER #

178	00F5H	15	ER12	BYTE ARRAY(15) DATA	185														
179	0104H	18	ER13	BYTE ARRAY(18) DATA	185														
180	0116H	11	ER14	BYTE ARRAY(11) DATA	185														
181	0121H	27	ER15	BYTE ARRAY(27) DATA	185														
182	013CH	18	ER16	BYTE ARRAY(18) DATA	185														
183	014EH	19	ER17	BYTE ARRAY(19) DATA	185														
184	0161H	32	ER18	BYTE ARRAY(32) DATA	185														
7			ERR.	LITERALLY	928														
185	0000H	38	ERRMSG	WORD ARRAY(19) DATA	193														
164	04BFH	147	ERROR.	PROCEDURE STACK=001CH		213	227	237	262	269	286	302	307						
				440 475 552 573 602 609 611 622 725 826 902 921 1055 1086															
				1093 1104															
164	0006H	1	ERRTYPE.	BYTE PARAMETER AUTOMATIC	165	193													
10	0098H	1	EXTSAVE.	BYTE	297	303													
30	0000H	1	F.	BYTE PARAMETER	31														
27	0000H	1	F.	BYTE PARAMETER	28														
24	0000H	1	F.	BYTE PARAMETER	25														
12	00A9H	1	F1	BYTE	238	652													
12	00AAH	1	F2	BYTE	239	653													
12	00ABH	1	F3	BYTE	240	654													
12	00ACH	1	F4	BYTE	241	655													
9			FALSE.	LITERALLY	218	413	458	499	565	585	712	729	730	731	736				
				740 846 880 1034 1035 1036 1089 1106															
701			FAST	LITERALLY	702														
11	0099H	1	FASTCOPY	BYTE	306	728	736	740	747										
151	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	152	153													
147	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	148	149													
143	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	144	145													
139	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	140	141													
10	0000H	36	FCB.	BYTE ARRAY(36) MEMBER(SOURCE)	10	216	234	238	239	240	241								
				242 243 245 246 247 297 303 304 664 665 799 802 803 805															
				833 853 1069 1073 1082															
104	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	105	106													
100	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	101	102													
96	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	97	98													
87	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	88	89													
74	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	75	76													
70	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	71	72													
112	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	113	114													
871	0000H	36	FCB.	BYTE ARRAY(36) MEMBER(FCBS)	883	936	957	976	1009										
119	0004H	2	FCB.	WORD PARAMETER AUTOMATIC	120	121													
3	0000H	33	FCB.	BYTE ARRAY(33) EXTERNAL(1)	816	817	826	1069											
10	0000H	36	FCB.	BYTE ARRAY(36) MEMBER(DEST)	10	223	228	650											
10	0000H	36	FCB.	BYTE ARRAY(36) MEMBER(ODEST)	627	645	646	650	652	653	654								
				655 656 657 658 659 789 832 833 936 837 853 1073 1082															
165	0000H	33	FCB.	BYTE BASED(FILEADR) ARRAY(33)	197	200													
108	0004H	2	FCBA	WORD PARAMETER AUTOMATIC	109	110													
870	008CH	2	FCBA	WORD PARAMETER	871	883	922	928	936	937	957	965	976	980					
				1008 1009															
871	0000H	38	FCBS	STRUCTURE BASED(FCBA)	883	922	928	936	937	957	965	976							
				980 1008 1009															
10	0090H	1	FEEDBASE	BYTE	511	512	521												
10	0091H	1	FEEDLEN.	BYTE	508	510	523	941											
7			FEXT	LITERALLY	202	223	791												
7			FEXTL.	LITERALLY	223														
7			FF	LITERALLY	394	399	408	416											
7			FILE	LITERALLY	1008	1062	1066	1077	1081	1088	1103	1105							

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P.O. BOX 575

PACIFIC GROVE, CA 92650

SER. =

164	0004H	2	FILEADR.	WORD PARAMETER AUTOMATIC	165	195	197	200
887	159CH	25	FILLQ.	PROCEDURE STACK=0008H	949	1002		
292	07A4H	164	FILLSOURCE	PROCEDURE STACK=0020H	452	749		
10	0095H	3	FILSIZE.	BYTE ARRAY(3)	216	245		
11	00ABH	1	FLEN	BYTE	883	890	930	931
7			FNSIZE	LITERALLY	199	788	932	999
9			FOREVER.	LITERALLY	79	507	567	707
212	0552H	15	FORMERR.	PROCEDURE STACK=0020H	854	868	1058	1063
17	013AH	1	FORMF.	BYTE AT	392			
7			FRSIZE	LITERALLY	10	222	871	931
7			FSIZE.	LITERALLY	165			
116	03EFH	15	GETDISK.	PROCEDURE BYTE STACK=0008H	1029			
487	0B87H	160	GETSOURCE.	PROCEDURE BYTE STACK=002CH	541	556	566	576
				603		578	581	588
434	0A76H	273	GETSOURCEC	PROCEDURE BYTE STACK=002BH	515			
17	013BH	1	GETU	BYTE AT				
123	0411H	15	GETUSER.	PROCEDURE BYTE STACK=0008H	1028			
353	131EH	37	GNC.	PROCEDURE BYTE STACK=0008H	862	896	901	908
				926	951	981	998	910
547	015EH	1	H.	BYTE	549	550	551	553
				581	587	588	593	599
547	015FH	1	HBUF	BYTE	580	592	599	
546	0C97H	241	HEXRECORD. . . .	PROCEDURE STACK=005AH	770			
17	013CH	1	HEXT	BYTE AT	769			
872	016BH	1	I.	BYTE	938	939		
700	016SH	1	I.	BYTE	732	733	735	737
972	016DH	1	I.	BYTE	975	976	978	
895	016BH	1	I.	BYTE	898	917	918	
874	016AH	1	I.	BYTE	876	877		
686	0164H	1	I.	BYTE	690	691	693	
343	0157H	1	I.	BYTE	349	350	351	353
391	0157H	1	I.	BYTE	401	403	404	405
165	0153H	1	I.	BYTE	199	200	202	
786	0166H	1	I.	BYTE	788	789	791	
17	013DH	1	IGNOR.	BYTE AT	586	769		
22	0000H		INPD	PROCEDURE BYTE EXTERNAL(4) STACK=0000H	462			
7			INPT	LITERALLY				
11	00A2H	1	INSPARC.	BYTE	259	306	750	765
972	01A7H	31	IO	BYTE ARRAY(31) DATA	976			
895	016CH	1	J.	BYTE	907	912	913	915
972	016EH	1	J.	BYTE	973	976	986	
251	015SH	1	J.	BYTE	279	280	281	282
972	016FH	1	K.	BYTE	974	980		
872	0167H	1	K.	BYTE	914	915		
17	013FH	1	KILDS.	BYTE AT	827	839		
7			LA	LITERALLY	875	1057		
			LAST	BUILTIN	876			
16	0134H	1	LASTUSER	BYTE	82	128		
7			LB	LITERALLY	875	962	981	990
547	0084H	2	LDA.	WORD	595			
887	0004H	1	LEN.	BYTE PARAMETER AUTOMATIC	888	890		
9			LF	LITERALLY	45	419	468	504
10	008FH	1	LINENO	BYTE	405	407	417	1040
7			LIT.	LITERALLY	7	701		
			LOW.	BUILTIN	290	614		
17	0140H	1	LOWER.	BYTE AT	483			
7			LPP.	LITERALLY	404			

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 101 S X 579
 PACIFIC GROVE, CA 93950

354 355
 SER =

7		LSTT	LITERALLY																	
423	0A5CH	26	LTRAN.	PROCEDURE	BYTE	STACK=0004H		434												
972			M.	LITERALLY		974														
11	007FH	1	MADE	BYTE		187 229 252 846 1035														
108	03C9H	19	MAKE	PROCEDURE	STACK=000AH		225													
489	0C27H	66	MATCH.	PROCEDURE	BYTE	STACK=0004H		519	529											
10	0092H	1	MATCHLEN	BYTE		491 497 498 523 524 941														
3	0000H	2	MAXB	WORD	EXTERNAL(0)		670 671													
13	00B0H	1	MAXLEN	BYTE	AT		57 58													
215	0561H	38	HAXSIZE.	PROCEDURE	BYTE	STACK=0002H		306												
	0000H		MEMORY	BYTE	ARRAY(0)		10 670 671 674													
24	0000H		HON1	PROCEDURE	EXTERNAL(5)	STACK=0000H		34	41	49	58	68	98							
						128 153 327 330 331 332 339 1022														
27	0000H		HON2	PROCEDURE	BYTE	EXTERNAL(6)	STACK=0000H		37	61	72	76	84							
						89 93 102 106 110 114 117 121 124 141 145 149 459 461														
30	0000H		HON3	PROCEDURE	WORD	EXTERNAL(7)	STACK=0000H		64											
155	049AH	37	MOVE	PROCEDURE	STACK=0008H		222 223 245 290 650 802 832 833													
						1014 1069 1073 1082														
11	00A4H	1	MULTCON.	BYTE		1015 1023 1041 1119														
783	1130H	248	MULTCOPY	PROCEDURE	STACK=0062H		1070													
2			MVERSION	LITERALLY		1021														
251	0082H	2	N.	WORD		254 265 276														
155	0004H	1	N.	BYTE	PARAMETER	AUTOMATIC		156	158											
734	008AH	2	NCOPIED.	WORD		812 825 841 845														
784	008BH	2	NDCN.	WORD		818 819 820 831														
10	007EH	2	NDEST.	WORD		254 290 334 336 337		450	614	680	757	762	8764	8778						
						1033														
11	00A1H	1	NENDCHD.	BYTE		758 774 1039 1090 1094 1115														
373	096DH	90	NEWLINE.	PROCEDURE	STACK=0040H		412													
784	0086H	2	NEXTDIR.	WORD		812 819 831														
440	0ABDH		NOTSOURCE.	LABEL		438 439 441														
10	007AH	2	NSBUF.	WORD		294 295 296 309 312 446 450 543 682 689 693 757														
						762 764 776 1033														
7			NSIZE.	LITERALLY		946 949 996														
10	007CH	2	NSOURCE.	WORD		248 294 443 447 454 455 543														
7			MULT	LITERALLY		1107														
17	0142H	1	NUMB	BYTE	AT		376 383 411 1097													
17	0143H	1	OBJ.	BYTE	AT		466 542 687													
10	0054H	38	ODEST.	STRUCTURE		134 222 328 623 626 627 645 646 647 648 650														
						652 653 654 655 656 657 658 659 660 789 832 833 836 837														
						853 1053 1055 1060 1066 1073 1077 1082 1085 1088 1095														
374	0153H	1	ONE.	BYTE		375 377														
70	030DH	19	OPEN	PROCEDURE	STACK=000AH		233 623													
702	0181H	26	OPTYPE	BYTE	ARRAY(26)	DATA	735 737													
19	0000H		OUTD	PROCEDURE	EXTERNAL(3)	STACK=0000H		329												
7			OUTT	LITERALLY																
17	0144H	1	PAGCNT	BYTE	AT		401 1100 1101													
1	0002H	614	PIPMOD	PROCEDURE	STACK=0064H															
5	0010H		PLM.	LABEL	PUBLIC		1014													
51	02BBH	16	PRINT.	PROCEDURE	STACK=0014H		192 632 637 842 1018													
362	092AH	40	PRINT1	PROCEDURE	STACK=0036H		370 371													
39	0235H	21	PRINTCHAR.	PROCEDURE	STACK=000AH		44 45 197 198 203 204 792 793													
						1043														
368	0952H	27	PRINTDIG	PROCEDURE	STACK=003CH		380 381 382													
47	02ABH	16	PRINTX	PROCEDURE	STACK=000AH		54 193 194 1025													
785	1278H	60	PRNAME	PROCEDURE	STACK=0012H		843													
7			PRNT	LITERALLY		1095														

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

PACIFIC GROVE, CA 93950

SER. ==

882	1577H	37	PUTCHAR	PROCEDURE STACK=0002H	891	934	950	1003											
315	0848H	137	PUTDCHAR	PROCEDURE STACK=002AH	345	347	356												
390	09C7H	123	PUTDEST	PROCEDURE STACK=0046H	556	575	591	592	593	615	772	1109							
				1112															
342	08D1H	89	PUTDESTC	PROCEDURE STACK=0030H	365	366	385	386	388	408	418								
11	00ASH	1	PUTNUM	BYTE 397 413 420 1037															
10	0093H	1	QUITLEN	BYTE 501 503 532 941															
17	0145H	1	QUITS	BYTE AT 527 529 531 610															
7			RB	LITERALLY 875 897															
562	0DDEH	23	RADDR	PROCEDURE WORD STACK=0056H		595													
36	0276H	15	RDCHAR	PROCEDURE BYTE STACK=0008H		473	633												
56	02CBH	20	RDCOM	PROCEDURE STACK=0008H	1044														
559	0DC4H	26	RDCS	PROCEDURE BYTE STACK=0050H		563	596	599	601										
540	0C69H	46	RDEF	PROCEDURE BYTE STACK=0030H		771													
555	0DB2H	18	RDHEX	PROCEDURE BYTE STACK=004AH		560													
139	045AH	16	RDRANDOM	PROCEDURE BYTE STACK=000AH		277	708												
112	03DCH	19	RENAME	PROCEDURE STACK=000AH	651														
147	047AH	16	REFSIZE	PROCEDURE BYTE STACK=000AH		244													
1030	0071H		RETRY	LABEL 210															
547	0160H	1	RL	BYTE 579 582 583 594 597	598														
12	00ADH	1	RO	BYTE 242 657															
			ROL	BUILTIN 234 627 803 837															
			ROR	BUILTIN 82															
17	0146H	1	RSYS	BYTE AT 234 837															
547	0162H	1	RT	BYTE 596															
155	0003H	2	S	WORD PARAMETER AUTOMATIC	156	157	159	160											
10	0006H	2	SBASE	WORD 10 296 309 454 670	674	674	691												
10	0000H	2	SDLEN	WORD 295 443 671 676 677															
10	0000H	1024	SBUFF	BYTE BASED(SBASE) ARRAY(1024)		296	309	454	691										
870	1364H	405	SCAN	PROCEDURE STACK=0024H	1053	1059	1116												
894	15B5H	248	SCANPAR	PROCEDURE STACK=0020H	963	982	991	1006											
87	0360H	22	SEARCH	PROCEDURE STACK=000EH	800	817													
92	037EH	21	SEARCHN	PROCEDURE STACK=000CH	808	821													
3			SEARFCB	LITERALLY 817 826 1069															
130	0436H	12	SETCUSER	PROCEDURE STACK=000EH	1050														
66	02F0H	16	SETDMA	PROCEDURE STACK=000AH	258	266	275	296	706	815									
133	0442H	12	SETDUSER	PROCEDURE STACK=000EH	186	221	257	619											
119	03FEH	19	SETIND	PROCEDURE STACK=000AH	647	660	666												
151	048AH	16	SETRANDOM	PROCEDURE STACK=000AH	264	305	705												
136	044EH	12	SETSUSER	PROCEDURE STACK=000EH	232	293	663	704	752	798	814								
220	0587H	77	SETUPDEST	PROCEDURE STACK=0020H	253														
685	0FCEH	63	SETUPEOB	PROCEDURE STACK=0002H	756														
231	05D4H	148	SETUPSOURCE	PROCEDURE STACK=0020H	746														
126	0420H	22	SETUSER	PROCEDURE STACK=000AH	131	134	137												
11	009EH	1	SFILE	BYTE 728 739 745 1038	1106	1108													
			SHL	BUILTIN 560 563 579 802 832															
			SHR	BUILTIN 370 670 671															
677	100DH	264	SIMPLECOPY	PROCEDURE STACK=005EH	847	1074	1083	1113											
669	0F65H	105	SIZEMEMORY	PROCEDURE STACK=002BH	744														
10	0003H	38	SOURCE	STRUCTURE 10 137 216 233 234 237 238 239 240 241 242															
				243 244 245 246 247 297 298 302 303 304 305 307 437 552															
				573 602 664 665 666 705 708 725 753 799 800 802 803 805															
				833 853 1059 1062 1069 1073 1077 1081 1082 1103 1105 1107 1111 1116															
10	0027H	2	SOURCER	WORD AT 710 717 722															
10	002BH	1	SOURCER2	BYTE AT 711 718 723															
11	00A3H	1	SPARFIL	BYTE 259 731															
7			SPECL	LITERALLY															

COPYRIGHT © 1981, 1982
DIGITAL RESOURCES
P.O. Box 579
PACIFIC GROVE, CA 93950

COPYRIGHT © 1981, VES
 LUNAR RESLON
 P.O. B. X 579
 PACIFIC GROVE, CA 93950

17	0147H	1	STARTS	BYTE AT	517	519	521	522	608	
12	00AEH	1	SYS.	BYTE	243	658				
7			TAB.	LITERALLY	344	388				
17	0148H	1	TABS	BYTE AT	346	350	351	353	1098	1099
10	0004H	2	TBLN.	WORD	689	691	693			
251	0080H	2	TDEST.	WORD	256	258	265	266	270	274 276 281 284 290
165	0154H	1	TEMP	BYTE	200	204				
9			TRUE	LITERALLY	79	217	229	308	420	436 494 507 567 584 707
					813	878	885	943	1030	1037 1038 1039
10	0025H	1	TYPE	BYTE MEMBER(ODEST)		328	1060	1066	1077	1085 1088 1095
10	0025H	1	TYPE	BYTE MEMBER(DEST)						
10	0025H	1	TYPE	BYTE MEMBER(SOURCE)		437	1062	1077	1081	1103 1105 1107 1111
871	0025H	1	TYPE	BYTE MEMBER(FCBS)		928	965	980	1008	
17	0147H	1	UPPER.	BYTE AT	481					
10	0024H	1	USER	BYTE MEMBER(ODEST)		134	853			
10	0024H	1	USER	BYTE MEMBER(DEST)						
126	0004H	1	USER	BYTE PARAMETER AUTOMATIC			127	128		
10	0024H	1	USER	BYTE MEMBER(SOURCE)		137	853			
871	0024H	1	USER	BYTE MEMBER(FCBS)		922	937			
422	0A42H	26	UTRAN.	PROCEDURE BYTE STACK=0004H			482	633	859	
17	014AH	1	VERIF.	BYTE AT	272					
2			VERSION.	LITERALLY	1016					
9			WHAT	LITERALLY	799	884	889			
250	0668H	316	WRITEDEST.	PROCEDURE STACK=0024H		335	449	618	681	763
143	046AH	16	WRITERANDOM.	PROCEDURE BYTE STACK=000AH			261			
17	0148H	1	WRROF.	BYTE AT	629					
17	014EH	1	ZEROP.	BYTE AT	479					
547	0163H	1	ZEROREC.	BYTE	565	571	584	585	586	
18	014FH	1	ZEROSUP.	BYTE	364	376				

MODULE INFORMATION:

CODE AREA SIZE = 16ADH 5805D
 CONSTANT AREA SIZE = 023AH 570D
 VARIABLE AREA SIZE = 0170H 368D
 MAXIMUM STACK SIZE = 0064H 100D
 1512 LINES READ
 0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. = _____